A TESTBED FOR COMPARING TRUST COMPUTATION ALGORITHMS

George Theodorakopoulos, and John S. Baras*
Institute for Systems Research,
University of Maryland,
College Park, MD 20742

ABSTRACT

Trust is the expectation of a person about another person's behavior. Trust is important for many security related decisions about, e.g., granting or revoking privileges, controlling access to sensitive resources and information, or evaluating intelligence gathered from multiple sources. More often than not, the issue is complicated even further because the person making the decision has no direct trust relationship with every single subject whose trustworthiness needs to be evaluated. So, the decision maker needs to rely on recommendations by others, and then somehow aggregate the trust related information that is collected. In this work we provide an algebraic framework in which we can describe multiple ways that trust related information can be aggregated to form a single value. We show the similarities and differences that the various so called *trust computation algorithms* have, and associate these with the algebraic properties of the framework that we consider.

1. INTRODUCTION

Trust is a weighted binary relation between two members of a network. As an example, consider a network of intelligence gathering agents, organized in a hierarchical manner. Trust could then be the expectation of a person A (presumably high in the hierarchy) that a person B (low in the hierarchy) is honest, as opposed, e.g., to being a double agent. The weight of this relation is then a way to quantify this expectation: The greater the weight, the higher the expectation.

Real life interactions build trust (or distrust) between some of the members of the network. In this way, what we call *direct* trust is created, and, since not all members of a network have direct interactions, such direct trust links do not exist between all pairs. However, members without direct interactions will also need to make trust assessments for others, as in our example above. Trust computation deals with the calculation of these *indirect* trust relations.

Ultimately, we want to combine all relevant direct trust relations and associated weights to come up with an indirect trust value (weight). For this, first of all we assume that trust is in some sense transitive: If A directly trusts B (to some degree) and B directly trusts C (to some degree), then we can derive how much A indirectly trusts C (through B). Hence, we can talk about *trust paths* from a source (A in this case) to a destination (C in this case). These paths can be of any length, not just of length 2, as in this case. A further observation is that there could be multiple trust paths from A to C, through nodes other than B, and all these paths will be relevant for the trust computation.

Several approaches to trust computation have been proposed in the literature by (Theodorakopoulos and Baras, 2006), (Jøsang, 1999), (Levien and Aiken, 1998), and others. Unfortunately, all these attempts have been made in a relatively ad-hoc fashion. With few exceptions, no researchers have compared their own approach to the others. As a result, someone – say, a network administrator – who believes that a notion of trust would be useful to incorporate in his administrative domain, has no easy way to choose which trust metric would be more suited to his needs.

It is this omission that we set out to correct in this paper. We show how multiple trust computation algorithms can be seen from a common viewpoint. Their properties are formalized within an algebraic framework. The benefit of this approach is that it is much easier to see the differences and similarities that exist. Moreover, it is easier to design an algorithm that satisfies the desired set of properties for a particular situation, since the relevant properties are clearly singled out. Finally, it is easier to implement and evaluate the algorithms under a common software solution which makes use of the common framework that all algorithms are instantiations of.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **01 NOV 2006**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE
A Testbed For Comparing Trust Computation Algorithms

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Systems Research, University of Maryland, College Park, MD 20742

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM002075.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**UU**	**8**	

The rest of the paper includes the description of our system model; detailed expositions of published algorithms under that model; requirements that the algorithms are intuitively expected to satisfy; and algebraic properties that the algorithms may or may not have. We interpret the algebraic properties in terms of practical implications. Finally, we emphasize proposals on evaluating the robustness of the algorithms to attacks by malicious adversaries.

2. GENERAL FRAMEWORK AND DESCRIPTION OF ALGORITHMS

We will be dealing with what is called *recommendation trust* in the literature, as opposed to *direct trust*. In other words, we assume that direct trust has already been built between some pairs of users in the network through real life interactions or otherwise, as mentioned earlier. Then, our area of interest is the combination of these direct trust values into indirect ones. The term *recommendation* comes from the fact that the intermediate trust values can be seen as recommendations of users for other users. Also note that, in general, the recommendation trust values of two users A and A' about a user B will differ. Different users can have different opinions about the same user.

Our model of this situation is a directed graph $G = (V, E)$ with weights on the edges, the weight function being $w : e \longrightarrow S, e \in E$. The set S contains all possible trust values, usually from some minimum to some maximum value. The nodes of the graph correspond to the users, and the edges and edge weights correspond to the direct trust relations and the degree of trust associated with each relation. The set of *neighbors* of a user i, denoted N_i, consists of all nodes $j \in V$ such that a directed edge $e = (i, j)$ exists. We distinguish a *source* node s. The task of the algorithm that we present is to compute the source node's opinions (recommendation trust values) for every other node (user). That is, we want to come up with a single value in S for each user in the network. We denote s's recommendation trust value for user d by $t(s, d) \in S$, where $t : V \times V \longrightarrow S$.

Each of the algorithms that we present differs in the values and interpretation of edge weights (the set S), and the way the function $t(s, d)$ is computed from the graph and the weights. The unifying theme is the path interpretation that can be given to these computations. More specifically, we can define two operators that can be used to combine the available direct trust information. The first operator, which we will call *concatenation* operator and denote with the symbol $\otimes$, is used to combine trust values along a path from the source to the destination, as shown in Figure 1. More formally, consider a path p_1 from the source s to the destination d comprising of the edges $e_1 = (s, a_1), e_2 = (a_1, a_2), \ldots, e_k = (a_{k-1}, d)$. The recommendation trust value of s about d along the path p_1 is

$$t^{p_1} = w(e_1) \otimes w(e_2) \otimes \ldots \otimes w(e_k). \qquad (1)$$

Note that it only makes sense to use the $\otimes$ operator for edge weights that are one after the other, i.e., form a directed path from the first to the last.

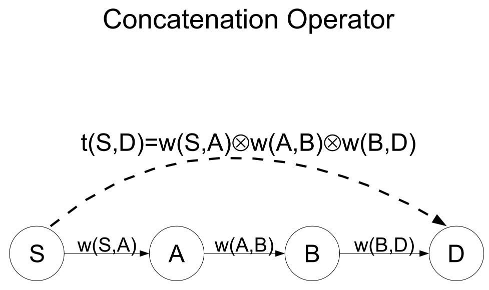

Figure 1: The concatenation operator $\otimes$ is used to combine opinions along a path.

The second operator, which we will call *summary* operator and denote with the symbol $\oplus$, is used to combine opinions computed along paths that start at the same node X, and end at the same node Y, i.e., paths that are, in a sense, parallel (see Figure 2). More formally, consider multiple paths $p_1, p_2, \ldots, p_n$ from the source s to the destination d with associated computed recommendation trust values $t^{p_1}(s, d), t^{p_2}(s, d), \ldots, t^{p_n}(s, d)$. The total recommendation trust is

$$t(s, d) = t^{p_1}(s, d) \oplus t^{p_2}(s, d) \oplus \ldots \oplus t^{p_n}(s, d). \qquad (2)$$

Overall, we can write

$$t(s, d) = \bigoplus_{\text{path } p: s \rightsquigarrow d} t^p(s, d). \qquad (3)$$

We now proceed to describe several trust computation algorithms within the framework that we have just built. For each algorithm, we will give the definition and interpretation of the weights, and the definition of the operators.

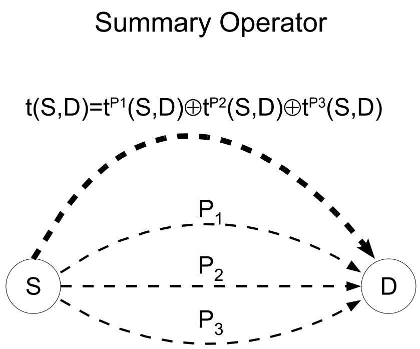

Figure 2: The summary operator $\oplus$ is used to combine opinions across paths.

2.1 Information Theoretic (Sun et al., 2006)

2.1.1 Entropy-Based

The weight is derived from a Bernoulli probability mass function, which, in effect, defines the probability p_{AB} that user B is trustworthy according to user A. The weight $w(A,B)$ is then computed as a function of the entropy of the Bernoulli distribution in the following way:

$$w(A,B) = \begin{cases} 1 - H(p_{AB}), & \text{for } 0.5 \leq p_{AB} \leq 1, \\ H(p_{AB}) - 1, & \text{for } 0 \leq p_{AB} \leq 0.5. \end{cases} \tag{4}$$

The entropy function is defined as $H(p) = -p\log_2 p - (1-p)\log_2(1-p)$, and since $0 \leq p \leq 1$, we can see that $-1 \leq w(A,B) \leq 1$. So, in this case, the set S is $S = [-1, 1]$.

The concatenation operator is:

$$w(A,B) \otimes w(B,C) = w(A,B)w(B,C), \tag{5}$$

i.e., regular multiplication.

The summary operator is:

$$\begin{aligned} t^{p_1}(s,d) \oplus t^{p_2}(s,d) = {} & \frac{w(e_1^{p_1})}{w(e_1^{p_1}) + w(e_1^{p_2})} t^{p_1}(s,d) + \\ & \frac{w(e_1^{p_2})}{w(e_1^{p_1}) + w(e_1^{p_2})} t^{p_2}(s,d), \end{aligned} \tag{6}$$

i.e., a weighted sum, where the weight of each path is proportional to the trust value on the first edge of the path.

2.1.2 Probability-Based

The weight $w(A,B)$ in this case is a pair of numbers (p_{AB}, σ_{AB}). The number p_{AB} is the mean of a Beta probability distribution function, and σ_{AB} is the variance of this Beta distribution. It is interpreted as the confidence that user A has about the trust value p_{AB}, i.e., how certain A is that p_{AB} is an accurate estimate of the probability that B is trustworthy.

As an aside, a Beta pdf is often used in the literature to model how direct trust values appear from direct positive and negative experiences between two users. The connection is that if user A has a positive and b negative experiences with user B, the mean of the associated Beta pdf will be $\frac{a}{a+b}$ and the variance will be $\frac{ab}{(a+b)^2(a+b+1)}$. This will help make it easier to understand why the summary operator does what it does. However, we do not look into what alternatives exist to the generation of direct trust values. Our aim, as we have noted, is to compare alternatives to the computation of recommendation trust values.

The concatenation operator is:

$$\begin{aligned} w(A,B) \otimes w(B,C) = {} & (p_{AB}, \sigma_{AB}) \otimes (p_{BC}, \sigma_{BC}) && (7) \\ = {} & (p_{ABC}, \sigma_{ABC}), && (8) \end{aligned}$$

where the two components are

$$p_{ABC} = p_{AB}p_{BC} + (1 - p_{AB})(1 - p_{BC}) \tag{9}$$

$$\begin{aligned} \sigma_{ABC} = {} & p_{AB}\sigma_{BC} + \frac{1}{12}(1 - p_{AB}) \\ & + p_{AB}(1 - p_{AB})(2p_{BC} - 1)^2 \end{aligned} \tag{10}$$

The summary operation is done through an intermediate transformation :

$$t^{p_1}(s,d) \oplus t^{p_2}(s,d) = (p_{sd}^{p_1}, \sigma_{sd}^{p_1}) \oplus (p_{sd}^{p_2}, \sigma_{sd}^{p_2}) \tag{11}$$

Each (p, σ) pair is transformed to an (a, b) pair. Then, the two pairs (a_1, b_1) and (a_2, b_2) are composed, and the result is transformed back to a (p, σ) pair.

$$(p,\sigma) \to (a,b) = \Big(p(\tfrac{p(1-p)}{\sigma} - 1), (1-p)(\tfrac{p(1-p)}{\sigma} - 1)\Big) \tag{12}$$

$$(a_1, b_1) \oplus (a_2, b_2) = (a_1 + a_2 - 1, b_1 + b_2 - 1) \tag{13}$$

$$(a,b) \to (p,\sigma) = \left(\frac{a}{a+b}, \frac{ab}{(a+b)^2(a+b+1)}\right) \tag{14}$$

2.2 EigenTrust (Kamvar et al., 2003)

The weights in this case are real numbers between 0 and 1: $S = [0,1]$. The weights are normalized on a user-per-user basis, i.e., for each user i, $\sum_{j \in N_i} w(i,j) = 1$.

The concatenation operator is:

$$w(A,B) \otimes w(B,C) = w(A,B)w(B,C), \tag{15}$$

i.e., regular multiplication.

The summary operator is:

$$t^{p_1}(s,d) \oplus t^{p_2}(s,d) = t^{p_1}(s,d) + t^{p_2}(s,d), \tag{16}$$

i.e., regular addition.

2.3 Probabilistic (Maurer, 1996)

In this case the weights are treated exactly as probabilities: $S = [0,1]$. A weight $w(A,B)$ is interpreted to be the probability that the directed edge (A,B) exists. Then, the recommendation trust value of user s for user d is equal to the probability that there exists at least one directed path from s to d. We assume throughout that the probabilities on different edges are independent.

The concatenation operator is:

$$w(A,B) \otimes w(B,C) = w(A,B)w(B,C), \tag{17}$$

i.e., regular multiplication.

The summary operator is:

$$\begin{aligned} t^{p_1}(s,d) \oplus t^{p_2}(s,d) = t^{p_1}(s,d) + t^{p_2}(s,d) \\ -t^{p_1}(s,d)t^{p_2}(s,d), \end{aligned} \tag{18}$$

which is derived from the simple law of the probability of the union of two events: $P(A \cup B) = P(A) + P(B) - P(A \cap B) = P(A) + P(B) - P(A)P(B)$.

2.4 Multi-level (Abdul-Rahman and Hailes, 1997)

The weights are discrete: $S = \{-1, 0, 1, 2, 3, 4\}$. The interpretation ranges from complete distrust (-1), to ignorance (0), to increasing levels of trust (1,2,3,4).

The concatenation operator is:

$$w(A,B) \otimes w(B,C) = \frac{w(A,B)}{4}\frac{w(B,C)}{4}, \tag{19}$$

i.e., multiplication of the values divided by 4. The division by 4 is presumably done to normalize the values to have a maximum equal to 1, but this is not explicitly stated in (Abdul-Rahman and Hailes, 1997).

The summary operator is:

$$t^{p_1}(s,d) \oplus t^{p_2}(s,d) = \frac{1}{2}t^{p_1}(s,d) + \frac{1}{2}t^{p_2}(s,d), \tag{20}$$

i.e., averaging. This operator can be applied to multiple opinions at once, taking the average of all of them:

$$\begin{aligned} t^{p_1}(s,d) \oplus t^{p_2}(s,d) \oplus \ldots \oplus t^{p^n}(s,d) = \\ \frac{1}{n}(t^{p_1}(s,d) + t^{p_2}(s,d) + \ldots + t^{p_n}(s,d)). \end{aligned} \tag{21}$$

2.5 Subjective Logic (Jøsang, 1999)

The weights are ordered triplets of positive real numbers that sum to 1: $S = (b,d,u), b+d+u = 1, b,d,u \in [0,1]$. These three numbers are called, respectively, *belief*, *disbelief*, and *uncertainty*.

The concatenation operator is:

$$\begin{aligned} w(A,B) \otimes w(B,C) &= (b_1,d_1,u_1) \otimes (b_2,d_2,u_2) \\ &= (b_1b_2, b_1d_2, d_1 + u_1 + b_1u_2). \end{aligned} \tag{22}$$

The summary operator is:

$$\begin{aligned} t^{p_1}(s,d) \oplus t^{p_2}(s,d) &= (b^{p_1}_{sd}, d^{p_1}_{sd}, u^{p_1}_{sd}) \oplus (b^{p_2}_{sd}, d^{p_2}_{sd}, u^{p_2}_{sd}) \\ &= \left(\frac{b^{p_1}_{sd}u^{p_2}_{sd} + b^{p_2}_{sd}u^{p_1}_{sd}}{k}, \frac{d^{p_1}_{sd}u^{p_2}_{sd} + d^{p_2}_{sd}u^{p_1}_{sd}}{k}, \frac{u^{p_1}_{sd}u^{p_2}_{sd}}{k} \right), \end{aligned} \tag{23}$$

where $k = u^{p_1}_{sd} + u^{p_2}_{sd} - u^{p_1}_{sd}u^{p_2}_{sd}$.

2.6 Path-strength (Lee et al., 2003)

The weights are real numbers between 0 and 1: $S = [0,1]$.

2.6.1 Strongest Path

The concatenation operator is the min operator:

$$w(A,B) \otimes w(B,C) = \min(w(A,B), w(B,C)). \tag{24}$$

The summary operator is the max operator:

$$t^{p_1}(s,d) \oplus t^{p_2}(s,d) = \max(t^{p_1}(s,d), t^{p_2}(s,d)). \tag{25}$$

2.6.2 Weighted Sum of Strongest Disjoint Paths

The concatenation operator is the min operator:

$$w(A,B) \otimes w(B,C) = \min(w(A,B), w(B,C)). \quad (26)$$

The summary operator is similar to the one in 2.1.1, a weighted sum of paths, where the weights are those of the first edges on each path. The difference is that now the paths are required to be disjoint.

2.7 Graph flows (Levien and Aiken, 1998)

This and the next algorithm are based on arguments related to flows in graphs.

In this algorithm, the edge weights are viewed as capacities. A unit flow is sent out from the source s, and the trust value for the destination d is equal to the fraction of the flow that reaches the destination. Edge weights are again between 0 and 1: $S = [0, 1]$.

We can define the concatenation and summary operators as follows:

The concatenation operator is the min operator:

$$w(A,B) \otimes w(B,C) = \min(w(A,B), w(B,C)). \quad (27)$$

The summary operator is regular addition:

$$t^{p_1}(s,d) \oplus t^{p_2}(s,d) = t^{p_1}(s,d) + t^{p_2}(s,d). \quad (28)$$

2.8 Certificate Insurance (Reiter and Stubblebine, 1999)

In this algorithm, the weights are nonnegative real numbers: $S = [0, \infty)$. Again, the algorithm is a flow algorithm, just as the previous one. The same operators apply. However, the interpretation of the weights is different. Here, weights are expressed in monetary terms, in particular dollars. An edge (i, j) corresponds to a public key certificate that user i has issued for the public key of user j. The weight $w(i, j)$ is the amount of dollars that i agrees to pay to someone who believes that i's certificate is correct and later it turns out that it was not. This amount of dollars can be thought of as a kind of insurance.

3. REQUIREMENTS FOR TRUST METRICS

So far, we have listed several pairs of concatenation and summary operators. All of them are used to do the same calculation, that is, compute the recommendation trust value that a source node s should place on a destination node d. Based only on our intuition about this objective we can derive some conclusions about the desirable behavior of the operators. In this section we discuss some conditions that these operators should satisfy.

First of all, a user should not be able to unilaterally increase the source's recommendation trust value for the destination to a level higher than the source's trust value for the user himself. In other words, if s's trust in user i is $t(s,i) \in S$, then $t^{P_i}(s,d) \preceq t(s,i)$, where $\preceq$ is a partial order defined on S, and P_i is the set of paths from s to d that pass through i. The rationale is to avoid maliciously manipulated reports of trust values by users. A user's trust values about others cannot be trusted more than the user himself. At the most, a user can give the maximum trust value to everyone else, but even then, we do not want the source to increase its own trust values for everybody beyond some level.

On a related note, if s knows about d only through i, i.e., the path $s \rightsquigarrow i \rightsquigarrow d$ is the only path from s to d, then s cannot trust d more than how much i trusts d. Simply, there is no reason for s to be more optimistic than i's recommendation is. Continuing from the last paragraph, $t^{P_i}(s,d) \preceq t(i,d)$. Translating these considerations into a condition for the concatenation operator, we impose that trust should decrease along a path.

$$a \otimes b \preceq a, b, \quad a, b \in S. \quad (29)$$

We now come to the summary operator, which deals with aggregating recommendation trust values derived from different paths. Throughout the literature there is a prevailing notion that more independent paths are better than fewer (Reiter and Stubblebine, 1998). More paths should in a sense be better than fewer paths, since more evidence is better than less evidence. It takes more malicious users to collaborate and subvert the trust computation algorithm, since at least one is needed on every path from the source to the destination.

But it is not the trust value that should increase;

it is the confidence in the accuracy of the computed value, as long as, and to the extent that the trust values of different paths agree. In other words, if all recommendations agree that the destination is untrustworthy, then it stands to reason that the result of the summary operator should not increase the trustworthiness of the destination.

However, this conclusion is not always correct. What happens if some recommendations are positive and some are negative? Then two results are possible: One is that the confidence in the accuracy of the computed value should decrease, since the recommendations are in conflict. The other one is that the computed value should be the average of the recommendations (possibly weighted by the respective confidence values), while the confidence in its accuracy should increase.

Which one is more correct depends on the particular situation. If trust is ultimately assumed to be binary, i.e, a user is in reality either fully trustworthy or fully untrustworthy, then the summary of conflicting opinions should decrease the confidence. This happens, for instance, when entities in the network are assumed to be divided in either friendly or enemy, with no gradation in between. On the other hand, trust can also be interpreted to be something that can legitimately be grey, as opposed to just black or white. It can be, for instance, the fraction of the time when a user has been seen to behave in a cooperative way, which is a quantity that can take any value between 0 and 1. In this case it is perfectly admissible to say that a user is trusted at a level of, say, 0.7. Therefore, two conflicting opinions could be reconciled by arguing that the two recommenders have seen different aspects of the behavior of the destination user. As a result, it would make sense to compute a summary value as the average of the two recommendations, and increase the total confidence value in the result.

So far, we have mentioned in passing a distinction between two concepts: trust and confidence. We defined trust to be an estimate of the behavior of a user, and confidence to be the accuracy of that estimate. However, not all the algorithms that we described incorporate the notion of confidence. As transpires from the discussion in this section, the usefulness of confidence is more apparent when there exist conflicting opinions. So, one situation that simplifies things is when conflicting opinions are not needed, or can be explicitly forbidden to exist. This saves us the trouble of having to deal with malicious users falsely accusing benign ones, but also prevents good users from notifying the network about potential misbehavior that they have noticed. If, for instance, trust values are used for access control decisions, then disallowing conflicting opinions amounts to disallowing revocation of privileges. Whether this can be tolerated or not depends on the particular situation.

4. ALGEBRAIC PROPERTIES OF THE OPERATORS

After going over the intuitive properties that we would like the operators to have, and the conditions under which we would like them to have those properties, we now proceed to the algebraic properties of these operators. The motivation for talking about algebraic properties is that they can be linked to issues with the numerical results that the computation returns. As a high level example to be elaborated on later, with certain operators it could happen that some edge weights are taken into account twice, which is clearly undesirable. Moreover, related to the algebraic properties are performance issues, as well as whether the computation can be done in a distributed manner or not.

First and foremost, both operators should be *closed* with respect to the set S, that is, if $a, b \in S$, then $a \otimes b \in S$, and $a \oplus b \in S$. The reason is that our admissible results are in the set S, and any value outside S has by definition no meaning in our computations. If S and $\otimes$ satisfy this property, then the pair $(S, \otimes)$ is called a *magma* or a *groupoid*. Similarly for the pair $(S, \oplus)$. Although this look like a fundamental property, some times it can be tricky to get right. For example, in one of the two proposed algorithms in (Sun et al., 2006), presented in Section 2.1.1, the summary operator is not closed for the set $S = [-1, 1]$. That summary operator is weighted averaging and it would create a problem in a situation where the following holds for four users A, B, C, and D:

$$w(A,B) = 0.95 \qquad w(B,C) = 1 \tag{30}$$

$$w(A,D) = -0.9 \qquad w(D,C) = 1 \tag{31}$$

The algorithm would then compute

$$\begin{aligned} t(A,D) &= t^{ABC}(A,C) \oplus t^{ADC}(A,C) \\ &= w(A,B)w(B,C) \oplus w(A,D)w(D,C) \\ &= \tfrac{w(A,B)}{w(A,B)+w(A,D)} w(A,B)w(B,C) \\ &\quad + \tfrac{w(A,D)}{w(A,B)+w(A,D)} w(A,D)w(D,C) \\ &= \tfrac{0.95}{0.95-0.9} 0.95 \cdot 1 + \tfrac{-0.9}{0.95-0.9}(-0.9 \cdot 1) \\ &= 34.25 \notin S = [-1,1] \end{aligned} \tag{32}$$

A property for the summary operators, which is so natural, that is satisfied by all summary operators described above, is the *commutativity* property:

$$a \oplus b = b \oplus a, \forall a, b \in S \tag{33}$$

Remembering that a and b correspond to recommendation values derived from paths, we can see that it would not make sense to differentiate between, in effect, the names of the two paths when combining them. Commutativity makes the pair $(S, \oplus)$ a *commutative magma.*

We do not need commutativity for the $\otimes$ operator. This makes sense, since reversing the order of the edges of a path need not necessarily result in the same outcome. However, we note that some of the concatenation operators proposed are indeed commutative (e.g. $\cdot$, min). This is not a problem, as long as the computation algorithm does not explicitly rely on the commutativity of this operator.

Another property of interest is associativity, and we would like both operators to be associative:

$$\begin{aligned} a \oplus (b \oplus c) =& (a \oplus b) \oplus c & (34) \\ a \otimes (b \otimes c) =& (a \otimes b) \otimes c, \forall a, b, c \in S & (35) \end{aligned}$$

The reasoning is that the order in which the operator is applied should not matter. The justification in the case of the summary operator is that, if we have computed a recommendation trust value based on the currently available information, and then another path appears, we want to be able to just “add” together the information from the new path, and not do the whole computation from scratch. This pertains to the efficiency of the computation, especially when done over a network where information will arrive with different delays. Unfortunately, one intuitive summary operator – namely, averaging – is not associative in general:

$$\exists a, b, c \in S : \text{avg}(a, \text{avg}(b, c)) \neq \text{avg}(\text{avg}(a, b), c) \tag{36}$$

The averaging operator is intended to average over all available paths simultaneously, and not to include them one by one. However, in distributed environments, this leads to the problems just stated. One way to overcome this problem would be to count the number of paths already taken into account in the current result, so as to weight properly the current result and the new path value.

When it comes to the concatenation operator, associativity means that the final result should depend only on the order with which the edges appear in the path, but not on the order with which we choose to do the calculations. Associativity is not satisfied by all the concatenation operators we have presented. For example, the operator presented in Section 2.1.2 is not associative. However, whether this is a significant problem or not depends on the actual computation algorithm and the way it is implemented distributedly.

As far as algebraic terminology goes, the pair $(S, \otimes)$ is now a *semigroup*, whereas the pair $(S, \oplus)$ is a *commutative semigroup.*

The last property we will consider is the *distributivity* (left and right) of $\otimes$ over $\oplus$:

$$\begin{aligned} a \otimes (b \oplus c) =& (a \otimes b) \oplus (a \otimes c) \\ (a \oplus b) \otimes c =& (a \otimes c) \oplus (b \otimes c) \end{aligned} \tag{37}$$

Distributivity is the most useful operation in terms of increasing the efficiency of computations. By inspection of (37), we see that the left sides need to compute two operations (one $\oplus$ in the parenthesis, and one $\otimes$ next). However, the right sides need three. This fact and the efficiency gains have been explored and discussed at length in the literature (Aji and McEliece, 2000). However, it seems to be the most difficult to satisfy.

We will just limit ourselves to discussing the flow-based metrics (presented in Sections 2.7 and 2.8) from the point of view of distributivity. The operators used there ($\otimes = \min, \oplus = +$) do not satisfy distributivity. For this reason, only if applied in a particular way will they return the correct (intended) result, which is the flow from the source to the destination. If all the paths from the source to the destination are edge-independent (share no common edges), then no particular way is needed. But if they are not independent, then the paths need to be decomposed into successive segments comprising parallel (edge-independent) subpaths and common edges (a *series-parallel* decomposition). Then the summary operator will be applied to the edge-independent segments and then the concatenation operator will be applied to the successive segments. This could be repeated as needed. However, this cannot be done with all graphs, so other methods for computing flows should be used.

In the case of Subjective Logic (Sec. 2.5) distributivity is also not satisfied. It seems that where the series-parallel decomposition cannot be done, there is no way to do the computation, so some graphs can simply not be handled by that algorithm.

If the two operators satisfy all the properties that we have assigned to them so far, then the triplet $(S, \otimes, \oplus)$ is an algebraic structure called a *semiring.*

5. ATTACK RESISTANCE

Levien and Aiken, in (Levien and Aiken, 1998), suggested a criterion for measuring the resistance of a trust metric to attackers. First, they distinguished between two types of attacks: node attacks, and edge attacks. Node attacks amount to a certain node being impersonated. So, the attacker can issue any number of arbitrary opinions (public key certificates in Levien's case) from the compromised node about any other node. Edge attacks are more constrained: Only one false opinion can be created per each attack. In other words, an attack of this type is equivalent to inserting a false edge in the trust graph. Obviously, a node attack is the more powerful of the two, since it permits the insertion of an arbitrary number of false edges.

The attack resistance of a metric can be gauged by the number of node or edge attacks that are needed before the metric can be manipulated beyond some threshold. For instance, it has been shown (Reiter and Stubblebine, 1999) that a single misbehaving entity (a 1-node attack) can cause the metric proposed in (Beth et al., 1994) to return an arbitrary result.

Here an important clarification has to be made: there are trust graphs that are "weaker" than others. When, for example, there exists only a single, long path between the source and the destination, then any decent metric is expected to give a low trust value. So, the attack resistance of a metric is normally judged by its performance in these "weak" graphs.

CONCLUSIONS

We have presented an algebraic framework that unifies many trust computation algorithms. By focusing on the algebraic properties of the algorithms under this framework, we are able to compare them in a much more rigorous way. We have shown links between the properties and considerations that can arise in practical implementations. As a result, we believe that a security practitioner can benefit from our exposition by adapting a particular metric to his own specifications, or even designing a new one. In the future, we will further pursue the formal evaluation of the resistance of metrics to attacks.

ACKNOWLEDGMENTS

This work is prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011. Research is also supported by the U.S. Army Research Office under grant No DAAD19-01-1-0494. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the U.S. Army Research Office.

REFERENCES

Abdul-Rahman, A. and Hailes, S., 1997: A distributed trust model. *Proc. of the 1997 New Security Paradigms Workshop*, 48–60.

Aji, S. M. and McEliece, R. J., 2000: The generalized distributive law. *IEEE Transactions on Information Theory*, **46**, 325–343.

Beth, T., Borcherding, M., and Klein, B., 1994: Valuation of trust in open networks. *ESORICS '94*, 3–18.

Jøsang, A., 1999: An algebra for assessing trust in certification chains. *Proc. of the Network and Distributed Systems Security Symposium*.

Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H., 2003: The EigenTrust algorithm for reputation management in p2p networks. *Proc. WWW2003*, 640–651.

Levien, R. and Aiken, A., 1998: Attack-resistant trust metrics for public key certification. *Proc. of the 7th USENIX Security Symposium*, 229–242.

Maurer, U., 1996: Modelling a public-key infrastructure. *ESORICS '96*, 325–350.

Reiter, M. K. and Stubblebine, S. G., 1998: Resilient authentication using path independence. *IEEE Trans. Comput.*, **47**, 1351–1362.

Reiter, M. K. and Stubblebine, S. G., 1999: Authentication metric analysis and design. *ACM Trans. Inf. Syst. Secur.*, **2**, 138–158.

Lee S., Sherwood, R. and Bhattacharjee, B., 2003: Cooperative peer groups in NICE. *Proc. IEEE Infocom 2003*, 1272–1282.

Sun, Y., Yu, W., Zhu, H. and Liu, K. J. R., 2006: A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks. *Proc. IEEE Infocom 2006*.

Theodorakopoulos, G. and J. S. Baras, 2006: On trust models and trust evaluation metrics for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, **24**, 318–328.